Evaluación de performance de engine 3D para dispositivos móviles

Federico Cristina [1], Sebastián Dapoto[1], Pablo Thomas[1], Patricia Pesado[1,2]

[1] Instituto de Investigación en Informática LIDI,
Universidad Nacional de La Plata – Argentina

[2] Comisión de Investigaciones Científicas de la Provincia de Buenos Aires – Argentina

{fcristina, sdapoto, pthomas, ppesado}@lidi.info.unlp.edu.ar

Resumen. En la actualidad existen diversos frameworks para el desarrollo de aplicaciones móviles 3D, con un denominador común en todos ellos: la performance es un aspecto crítico incluso más decisivo que en los equipos de escritorio, los cuales en general cuentan con mayor capacidad de cómputo. Las herramientas de análisis de rendimiento o *profiling* con las que cuentan estos frameworks pueden ayudar en cierta medida a determinar la posible existencia de cuellos de botella en la ejecución de aplicaciones. Sin embargo, este tipo de herramientas tiene ciertas limitaciones tales como abarcar solo un espectro de las posibles causas del problema o acotar el análisis a ciertos escenarios en particular. El presente artículo propone una evaluación y medición de la incidencia de las principales características en la performance de las aplicaciones móviles 3D.

Palabras Clave: Unity engine, dispositivos móviles, aplicaciones 3D, performance

1 Introducción

En la actualidad los dispositivos móviles son cada vez más sofisticados y su evolución tecnológica permite ejecutar aplicaciones cada vez más complejas y con rigurosos requerimientos de hardware.

Sin embargo, cuando estas aplicaciones son visuales e incluyen gráficos tridimensionales, es posible notar cierta degradación en la fluidez de ejecución. Esta pérdida en la eficiencia de ejecución es debida a las características propias del dispositivo en el cual se ejecuta la aplicación, pero también a las características de la herramienta de desarrollo y/o de su implementación.

El número de frameworks para la creación de aplicaciones móviles 3D interactivas es cada vez mayor. Cada uno de estos frameworks posee diferentes características que los hacen adecuados para distintos tipos y magnitudes de proyecto.

Al elegir un framework en particular es posible basarse en diversos criterios tales como: la comunidad existente, los lenguajes de codificación permitidos, la facilidad

de uso, la calidad de los gráficos 3D resultantes, entre otros. Sin embargo, uno de los principales puntos de interés es obtener una buena performance en cuanto a visualización y fluidez.

En dispositivos móviles toma mayor relevancia la limitada capacidad de procesamiento comparada con – por ejemplo – la de los equipos de escritorio. Por este motivo, es importante conocer las capacidades y limitaciones del framework a utilizar en lo que respecta a la performance de visualización.

A fin de lograr este objetivo, el presente trabajo propone una evaluación que permite aislar, analizar y dimensionar la incidencia de las principales características relacionadas con la performance visual de las aplicaciones 3D móviles.

Esta evaluación brinda un soporte al ingeniero de software de aplicaciones móviles 3D, posibilitando la identificación de los factores que ralentizan la performance visual de las aplicaciones que desarrolla, y permitiendo ajustar estos puntos críticos hasta lograr la fluidez deseada.

El resto del paper se organiza del siguiente modo: la sección 2 describe la motivación del análisis propuesto; la sección 3 presenta en detalle la evaluación; la sección 4 exhibe la experimentación realizada y, finalmente, en la sección 5 se exponen las conclusiones y el trabajo a futuro.

2 Motivación

La gestación del análisis de performance que propone este trabajo se inicia durante el proceso de desarrollo de dos aplicaciones móviles: R-Info3D [1] e InfoUNLP3D [2]. La primera herramienta es un entorno 3D de aprendizaje de algoritmia básica, mientras que la segunda es un escenario virtual 3D de la Facultad de Informática de la Universidad Nacional de La Plata. Ambos proyectos resultaron en aplicaciones inmersivas, las cuales por su naturaleza presentan un elevado costo computacional, principalmente en lo que refiere a visualización [3].

Tanto en el proceso de implementación como en su posterior ejecución en distintos dispositivos móviles, se encontraron limitaciones en cuanto a la performance obtenida. Se procedió a reconocer los puntos críticos que incidían en la fluidez visual y se determinaron ciertos umbrales que no debían ser superados. Este análisis llevó a modificar las aplicaciones para lograr un mejor funcionamiento en los diferentes tipos de dispositivos móviles.

Previamente al inicio del desarrollo de las herramientas mencionadas, se realizó un análisis de los principales engines de uso libre para el desarrollo de aplicaciones móviles 3D. Los engines considerados fueron: Unity [4], uno de los más populares y sencillos de utilizar; Unreal Engine [5], en comparación con otros engines, algo más complejo de usar y con requerimientos de hardware más elevados; CryEngine [6], pensado principalmente para desarrollos 3D en primera persona, su instalación y uso no son procesos triviales.

R-Info3D e InfoUNLP3D fueron desarrolladas con el framework Unity. Dado que no existe un framework que sea óptimo en todos los aspectos posibles, la elección de Unity por sobre el resto de los frameworks de desarrollo móvil 3D analizados estuvo basada en una serie de factores, entre los que se destacan:

- Tutoriales: existen una gran cantidad de tutoriales y ejemplos que guían al nuevo usuario en el proceso de aprendizaje del uso del framework. Estos tutoriales están categorizados según el tipo de desarrollo, son audiovisuales y están provistos de todos los elementos necesarios para realizarlos (objetos, audios, imágenes, scripts, etc).
- Documentación: el manual de usuario [7] es amplio, de fácil comprensión y está subdividido debidamente en distintas categorías. Contiene un buscador que facilita la búsqueda de un tema o funcionalidad en particular.
- Requerimientos de software/hardware: los requisitos de sistema son considerablemente más bajos en comparación con otros frameworks similares. Esto, acompañado de una instalación simple y rápida, incentiva al usuario/desarrollador a comenzar rápidamente con la utilización del framework.
- Componentes disponibles: cuenta con un repositorio (*Asset Store*) en donde es posible encontrar una gran variedad de componentes que pueden acelerar el desarrollo de las aplicaciones. Contiene un buscador de componentes que permite realizar búsquedas complejas mediante distintos tipos de filtro.
- Facilidad de aprendizaje / uso: el framework es versátil y permite trabajar en dos lenguajes diferentes: C# y javascript. Esto, sumado a la documentación, los tutoriales, el repositorio y la comunidad existente, generan un escenario ideal para aprender rápidamente a usar el framework y resolver cualquier problema que pueda surgir en el proceso.
- Comunidad: la gran popularidad de Unity es un factor preponderante al momento de elegirlo. Es el framework 3D más utilizado y su comunidad está compuesta por más de dos millones de usuarios [8]. Contiene un foro subdividido en categorías en donde es posible plantear las situaciones o problemas que pueden surgir durante el desarrollo de una aplicación.

Existen diferentes propuestas de análisis de performance de engines 3D para aplicaciones móviles. Una de estas propuestas [9] evalúa la performance de las aplicaciones teniendo en cuenta el consumo de CPU y/o GPU que éstas generan. Otros trabajos [10, 11] basan su evaluación en un análisis de las principales características y funcionalidades de los frameworks, dando como resultado una lista o tabla comparativa.

Ninguno de los trabajos encontrados adopta el enfoque aquí propuesto, es decir, que den algún tipo de guía para el desarrollo de aplicaciones móviles 3D basándose en la performance final de visualización lograda en dichas aplicaciones.

3 Evaluación propuesta

El objetivo principal de la evaluación propuesta consiste en aislar cada una de las principales características que abarca una aplicación 3D; especialmente aquellas que tienen impacto directo en la performance, tiempo de respuesta y fluidez de ejecución de las aplicaciones generadas con el engine [12].

Características tales como el número de polígonos, la aplicación de luces y sombras, la utilización de texturas y/o transparencias, la visualización de sistemas de partículas y el cálculo de la física de objetos que componen la escena, son ejemplos de los principales ítems a evaluar.

Si bien la incidencia de estas características sobre la performance varía de acuerdo al software y al hardware sobre el cual se ejecuta la aplicación, las pruebas realizadas demuestran que existe un patrón común respecto a la degradación de performance en relación al aumento de los requerimientos visuales.

En función del conjunto de características previamente mencionado, se define una serie de pruebas independientes para evaluar la performance. Dichas pruebas se enumeran a continuación:

1. Renderización de objetos simples (*Basic Mesh Rendering*): se presentan progresivamente en pantalla objetos simples sin textura en movimiento en una escena sin iluminación ni sombras. Los objetos deben rotar continuamente a velocidad constante. El número de objetos en pantalla irá creciendo acorde transcurre el tiempo. Se contabiliza el número de fotogramas por segundo o *frame rate* (FPS) a lo largo de la simulación en función del número de objetos.
2. Renderización de objetos complejos (*Complex Mesh Rendering*): consiste en visualizar un objeto complejo en movimiento, el cual debe contener un elevado número de polígonos. La distancia de renderizado (*clipping plane*) se va incrementando a medida que avanza la prueba. Se contabiliza el FPS a lo largo de la simulación en función de la distancia de renderizado.
3. Luces y sombras (*Lights & Shadows*): se realiza una simulación similar a la renderización de objetos simples, pero en este caso la escena contiene iluminación y objetos con proyección y recepción de sombras. Se contabiliza el FPS en función del número de objetos.
4. Texturas (*Textures*): se realiza una simulación similar a la renderización de objetos simples, pero en este caso los objetos poseen texturas complejas, como puede ser transparencias, reflejos, etc. Se contabiliza el FPS en función del número de objetos.
5. Sistemas de partículas (*Particle Systems*): se crea una escena en donde se presenta progresivamente nuevas instancias de un sistema de partículas (por ejemplo humo, chispas, explosión, etc.). Se contabiliza el FPS en función del número de sistemas de partículas.
6. Física (*Physics*): se realiza una simulación similar a la renderización de objetos simples, pero en este caso a los objetos se les aplica reglas de física, como por ejemplo *gravedad*. Se contabiliza el FPS a lo largo de la simulación en función del número de objetos.

Aplicando este conjunto de pruebas se simplifica considerablemente la tarea de determinar los puntos críticos a optimizar en las aplicaciones desarrolladas, posibilitando así la identificación del punto de equilibrio en la calibración de las características analizadas.

En particular, para el caso de estudio de este trabajo se utilizaron cubos como objetos simples, mientras que para la prueba de renderización de un objeto complejo se eligió el modelo del edificio de la Facultad de Informática, que es utilizado en la aplicación InfoUNLP3D. Dicho objeto contiene más de 500.000 polígonos y un

elevado número de ventanas. Por esto último, para las pruebas de texturas se aplicó transparencia de tipo vidrio a los elementos. En cuanto al sistema de partículas, se generó uno similar al utilizado por el robot en R-Info3D al momento de ejecutar una instrucción de reposicionamiento (Pos).

4 Experimentación

El experimento consistió en realizar las pruebas enumeradas en el punto anterior en un conjunto de dispositivos móviles con diferentes características. Los dispositivos utilizados fueron tres smartphones y dos tablets: Samsung Galaxy S2 (smartphone), Samsung Galaxy J5 (smartphone), LG L5 II (smartphone), Asus MemoPad FHD10 (tablet) y Acer B1-730 (tablet). Estos dispositivos presentan diferencias considerables en sus prestaciones y cuentan con diferentes arquitecturas de hardware, como ARM y x86.

Las figuras 1 y 2 muestran los valores medios de los resultados obtenidos en las pruebas realizadas a lo largo de todos los dispositivos, considerando los dos extremos posibles en lo que respecta a calidades de renderizado en la plataforma Unity: *Fastest* (la más baja) y *Fantastic* (la más alta) [13].

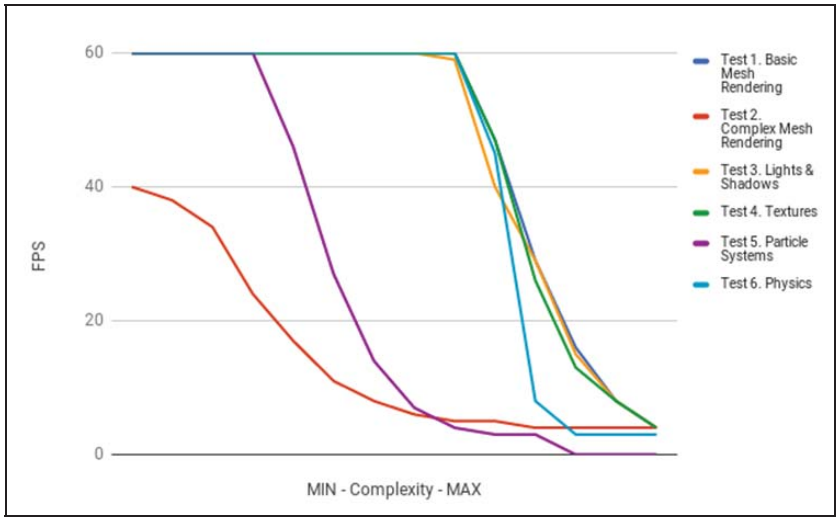

Fig. 1. FPS. Evolución de cada prueba en calidad *Fastest*.

En todos los casos, la información se encuentra normalizada en función del rango de valores mínimos y máximos a renderizar dependiendo del tipo de prueba realizada. En las pruebas 1, 3, 4, 5 y 6 el número de objetos en pantalla inicia en 1 y se incrementa gradualmente a lo largo de la simulación hasta llegar a un valor elevado, como puede ser 10000 objetos. Para la prueba 2, la evolución en este caso es la distancia de renderizado, que puede ir desde un mínimo de 200 hasta un máximo de

2000 unidades de distancia. Adicionalmente, el *frame rate* ideal considerado es de 60 FPS.

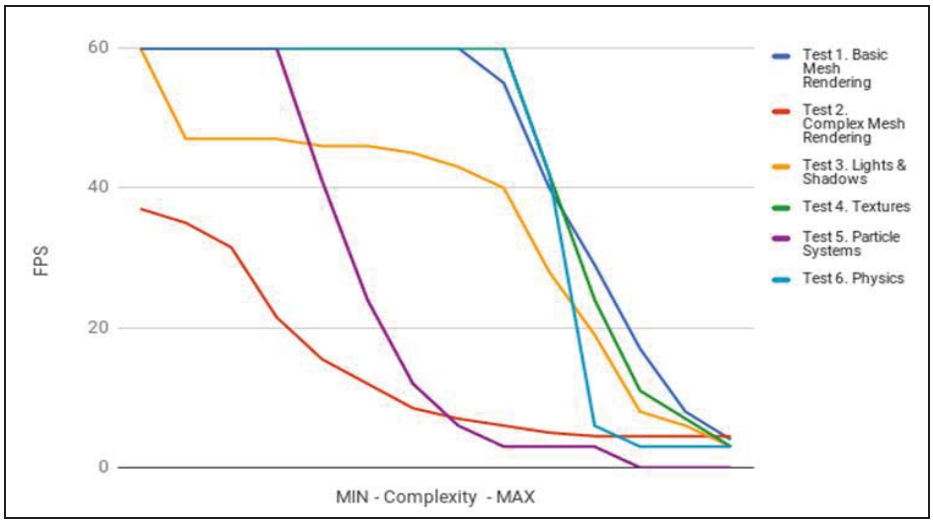

Fig. 2. FPS. Evolución de cada prueba en calidad *Fantastic*.

Los resultados de las pruebas permiten obtener un conjunto de conclusiones de interés, posibilitando la optimización correspondiente de las aplicaciones. Las observaciones a continuación son el resultado de aplicar la evaluación propuesta específicamente para el caso de estudio en cuestión.

- El render básico tanto en *Fastest* como en *Fantastic* no presenta diferencias en su rendimiento. Ambas curvas presentan prácticamente la misma evolución a lo largo de la simulación.
- La luz afecta la performance en calidad *Fantastic* pero no así en *Fastest*, en donde parece ser directamente ignorada en lo que respecta a cálculos y renderización. Tal como se detallará posteriormente, la aplicación de luces/sombras es uno de los factores determinantes en lo que respecta a performance visual.
- El número de polígonos en el modelo de la Facultad de Informática (+500k) es excesivamente alto para la capacidad de procesamiento de los dispositivos móviles utilizados, a fin de que la simulación pueda ser visualizada con la fluidez necesaria. Bajo ambas calidades gráficas la degradación es prácticamente igual.
- El impacto de aplicación de texturas no parece afectar considerablemente la performance general, al menos para la textura utilizada (la cual incluye transparencia), aplicada mediante un tipo de sombreado estándar, más conocido como *standard shader*, y un tamaño de textura moderado.
- Los sistemas de partículas, al ser 2D no se ven considerablemente afectados por los cambios de la calidad; pero ya un volumen alto de los mismos impide

la correcta ejecución de la simulación dada la gran cantidad de cálculos individuales que requiere cada una de las partículas que componen los sistemas.

- La aplicación de detección de colisiones/leyes de física tiene una incidencia relativamente menor sobre la performance cuando se trata de un número de objetos en escena "razonable" (por ejemplo, menos de 2000 cubos).

La figura 3 muestra la relación de FPS entre las calidades *Fastest* y *Fantastic*. Una relación igual a 1 denota que la prueba en cuestión incide de la misma forma en ambas calidades; mientras que un valor positivo denota mayor incidencia sobre *Fantastic* y un valor negativo mayor incidencia sobre *Fastest*. En la prueba 3 (luces y sombras) puede apreciarse notoriamente la mayor incidencia sobre la performance utilizando la configuración *Fantastic*. Se observa una fluidez general promedio del 50% con respecto a la configuración *Fastest*, que obviamente relega calidad gráfica en cuanto a estas características.

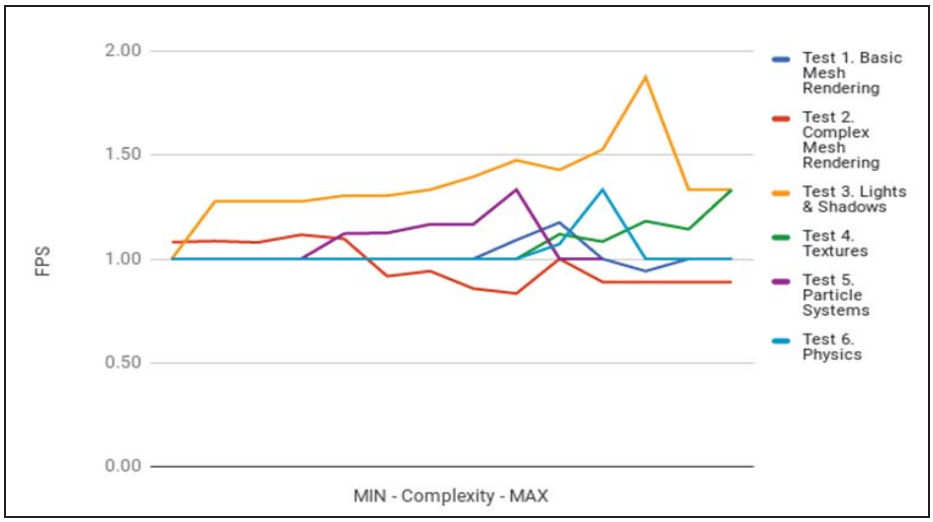

Fig. 3. FPS. *Fastest* vs. *Fantastic*.

Por otra parte, tal como se detalló previamente, la complejidad del objeto a ser renderizado (prueba 2) impacta equitativamente tanto en calidad *Fastest* como en *Fantastic*, y superado cierto umbral la distinción entre una renderización y otra es prácticamente nula o incluso negativa.

Gracias a este análisis, fue posible lograr un equilibrio aceptable entre calidad gráfica y fluidez de visualización en las aplicaciones desarrolladas en Unity. La figura 4 presenta una escena de la aplicación InfoUNLP3D con el mayor nivel posible de detalle en características tales como texturas, luces y sombras, pero con una performance extremadamente pobre en lo que respecta a FPS. La figura 5 muestra la escena una vez modificada a fin de lograr una fluidez óptima, aunque sacrificando en exceso las características previamente mencionadas. La figura 6 presenta los

resultados logrados mediante la calibración óptima seleccionada, basada en los resultados obtenidos en las pruebas. En dicha calibración se prioriza tanto la calidad de imagen como la fluidez de visualización. Se omite, por ejemplo, el uso de luces focalizadas (*spot lights*) para evitar la excesiva generación de sombras, dado que la finalidad de la aplicación no es presentar un render fotorealístico sino servir como guía de referencia para los alumnos.

Fig. 4. InfoUNLP3D. Máximo nivel de detalle.

Fig. 5. InfoUNLP3D. Mínimo nivel de detalle.

Bajo este contexto, se agregó además una nueva funcionalidad a la aplicación InfoUNLP3D que permite al usuario configurar la distancia de renderizado (y por consiguiente el número de polígonos en escena). Al evitar la renderización de los elementos más distantes, se gana aún mayor fluidez sin perder la funcionalidad de la

herramienta, que se basa en recorrer la facultad recibiendo información de las aulas cercanas. Esta nueva opción permite utilizar la aplicación en dispositivos móviles con menor capacidad de procesamiento.

Fig. 6. InfoUNLP3D. Calibración óptima seleccionada.

5 Conclusiones y trabajo a futuro

Este trabajo propone un conjunto de heurísticas para determinar los principales factores de incidencia en la degradación de la fluidez y visualización de las aplicaciones móviles 3D.

La evaluación propuesta permite individualizar estos factores a fin de rediseñar una aplicación y reducir considerablemente sus problemas de performance.

Como prueba de concepto y para validación de su efectividad, se aplicó la evaluación sobre el engine Unity. Se llevaron a cabo las pruebas de simulación, obteniendo como resultado final la información necesaria para realizar las optimizaciones correspondientes en las aplicaciones desarrolladas con el engine mencionado.

A partir de este trabajo, un ingeniero de software de aplicaciones móviles 3D, dispone de un conjunto de pautas a considerar en la optimización de la performance visual de las aplicaciones desarrolladas.

A futuro se prevé aplicar esta misma evaluación sobre otros engines de desarrollo de aplicaciones móviles 3D, tales como Unreal Engine y CryEngine; e idealmente realizar una comparación de performance entre los frameworks para cada una de las características evaluadas.

Referencias

1. Cristina, F.; Dapoto, S.; Thomas, P.; Pesado, P. “Prototipo móvil 3D para el aprendizaje de algoritmos básicos”. XXI Congreso Argentino de Ciencias de la Computación CACIC. 2015. ISBN: 978-987-3724-37-4.
2. Cristina, F.; Dapoto, S.; Thomas, P.; Pesado, P. “InfoUNLP3D: An interactive experience for freshman students”. XXII Congreso Argentino de Ciencias de la Computación CACIC. 2016. ISBN: 978-987-733-072-4.
3. Linowes J. “Unity Virtual Reality Projects”. 2015. ISBN-13: 978-1783988556
4. Unity. https://unity3d.com
5. Unreal Engine. https://www.unrealengine.com/
6. CryEngine. https://www.cryengine.com/
7. Unity online manual. http://docs.unity3d.com/Manual/index.html
8. Unity Blogs: The Unity Community. https://blogs.unity3d.com/2014/03/11/we-got-karma/
9. Messaoudi F., Simon G., Ksentini A. "Dissecting Games Engines: the Case of Unity3D". International Workshop on Network and Systems Support for Games (NetGames). 2015. Electronic ISSN: 2156-8146.
10. Akekarat Pattrasitidecha. "Comparison and evaluation of 3D mobile game engines". Chalmers University of Technology. University of Gothenburg. 2014.
11. Petridis P., Dunwell I., Panzoli D., Arnab S., Protopsaltis A., Hendrix M., de Freitas S. "Game Engines Selection Framework for High-Fidelity Serious Applications". "International Journal of Interactive Worlds". 2012. Article ID 418638. DOI: 10.5171/2012.418638.
12. Optimizing Graphics Performance. https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html
13. Unity Manual: Quality Settings. https://docs.unity3d.com/Manual/class-QualitySettings.html